SchemaAnalyst

Search-based Testing for Relational Database Schemas

Cody Kinneer

Institute for Software Research
Carnegie Mellon University

Additional Co-Authors: Phil McMinn, Chris J. Wright, Cody Kinneer, Colton McCurdy, Michael Camara, and Gregory M. Kapfhammer

Relational Databases

Databases are everywhere!

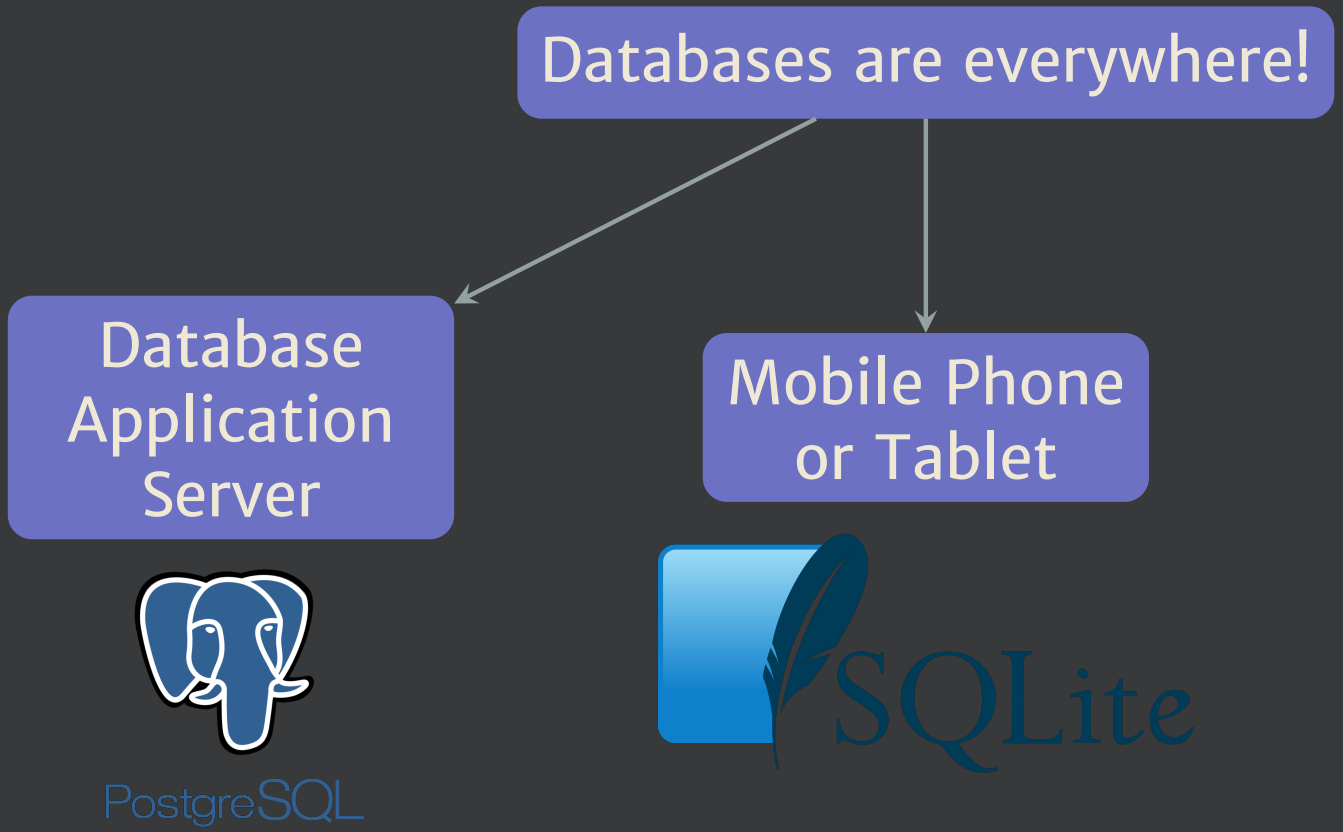
Relational Databases

Databases are everywhere!

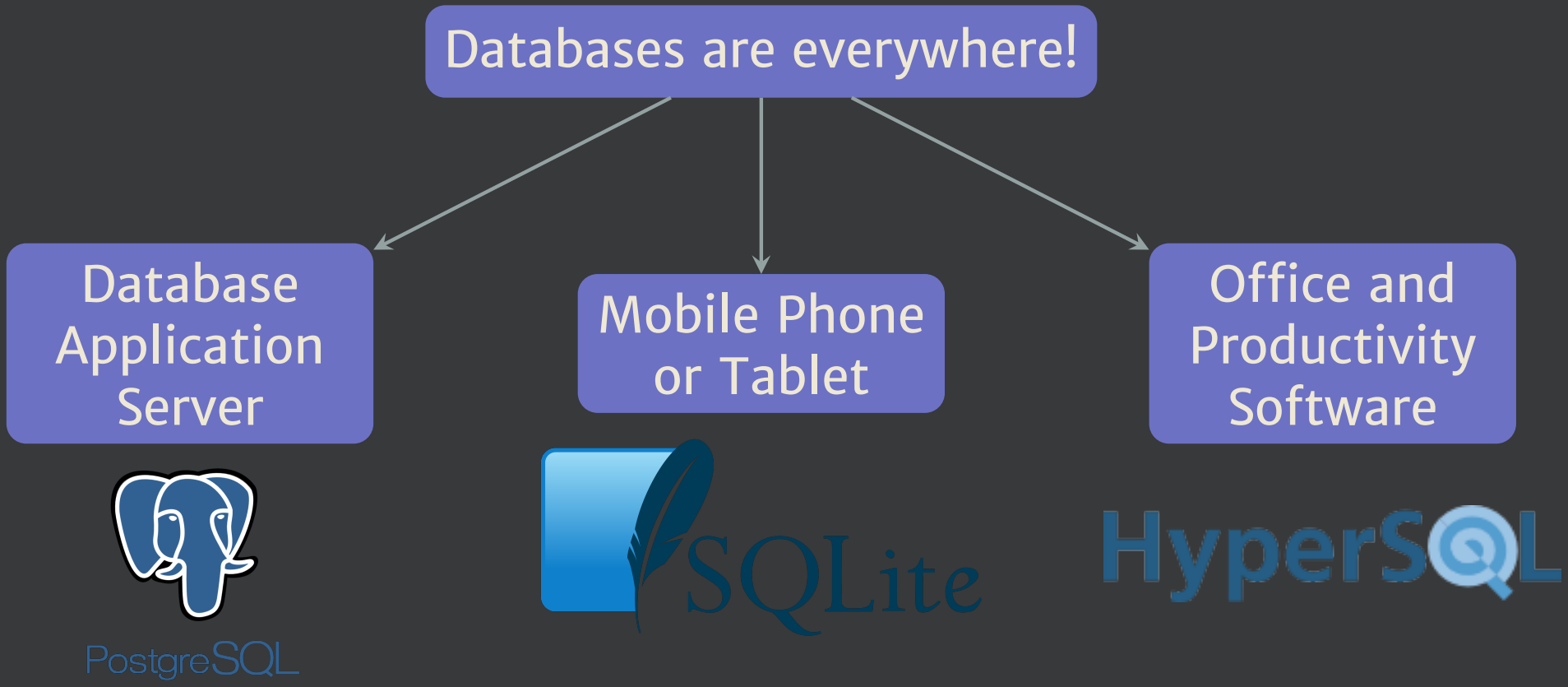
Database Application Server

PostgreSQL

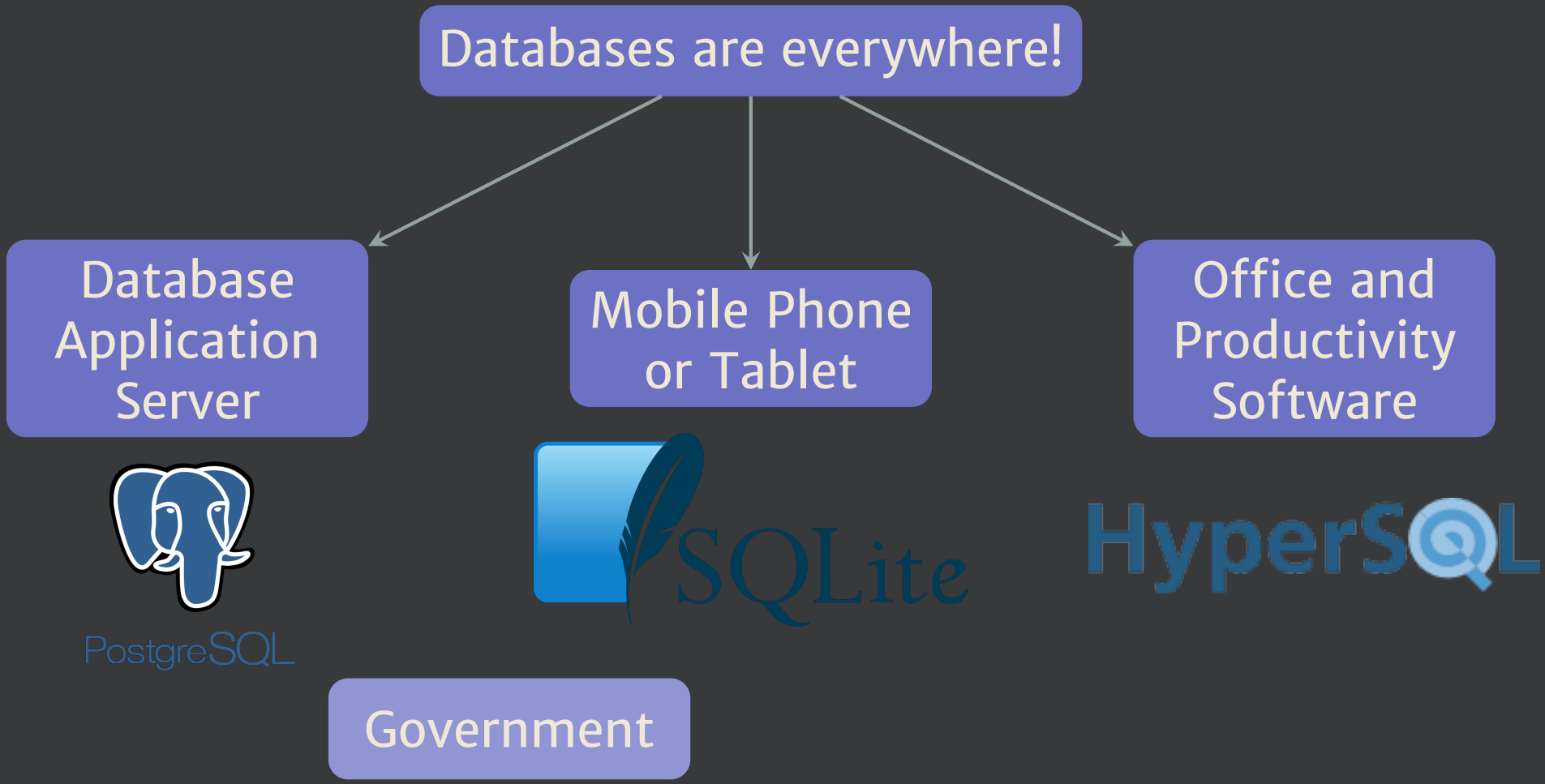
Relational Databases



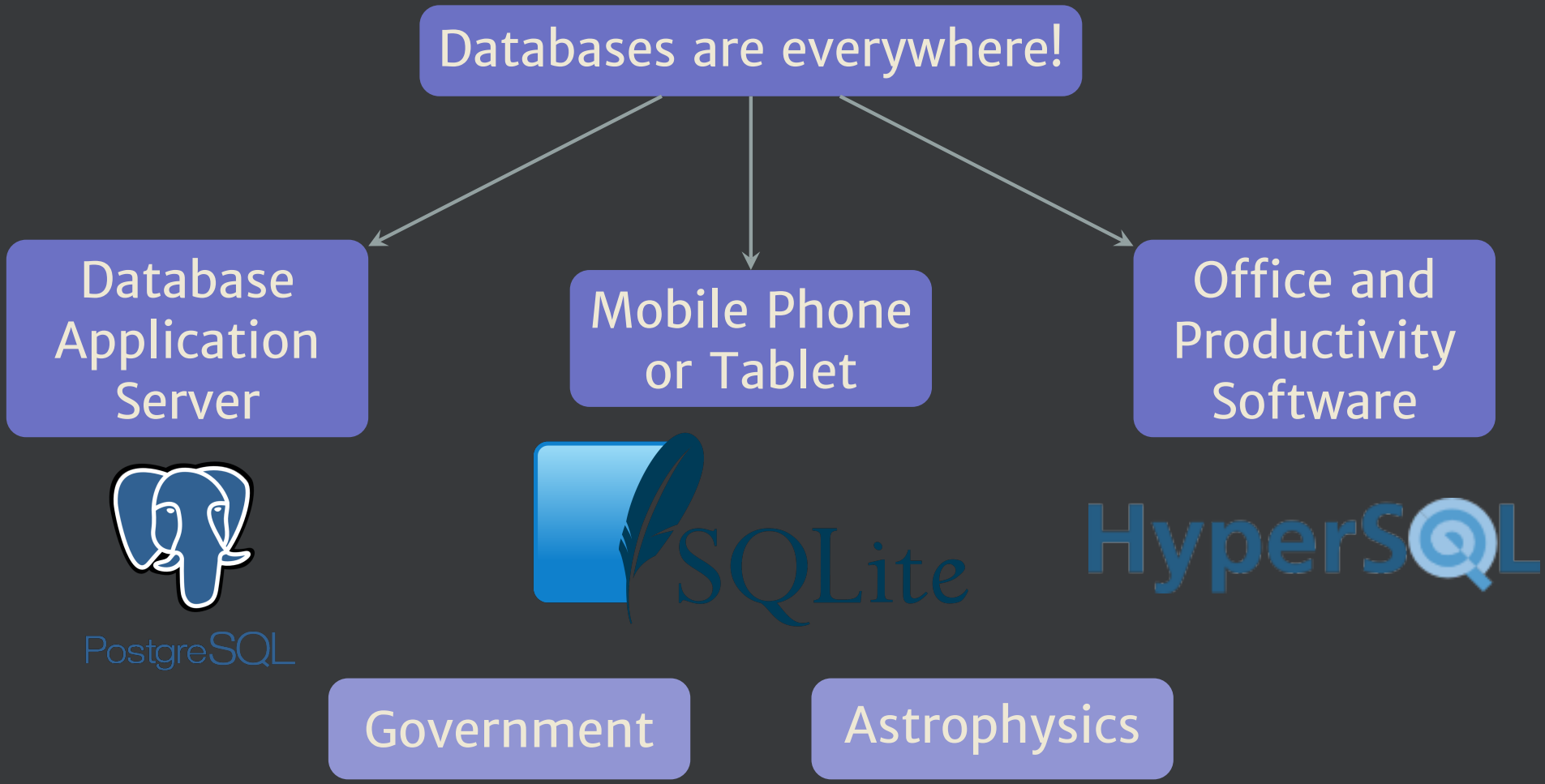
Relational Databases



Relational Databases



Relational Databases



Relational Databases

Databases are everywhere!

Database Application Server

Mobile Phone or Tablet

Office and Productivity Software

PostgreSQL

HyperSQL

Government

Astrophysics

Over 1,000,000 posts!

Database Schemas

```
1  CREATE TABLE DEPT_INFO (
2   DEPT_ID INTEGER NOT NULL,
3   DEPT_NAME VARCHAR(50),
4   PRIMARY KEY (DEPT_ID)
5  );
6
7  CREATE TABLE OFFICE_INFO (
8   OFFICE_ID INTEGER NOT NULL,
9   OFFICE_NAME VARCHAR(50),
10  HAS_PRINTER SMALLINT,
11  PRIMARY KEY (OFFICE_ID)
12 );
```

Figure: A sample of the UnixUsage schema.

Database Schemas

```
1  CREATE TABLE DEPT_INFO (
2   DEPT_ID INTEGER NOT NULL,
3   DEPT_NAME VARCHAR(50),
4   PRIMARY KEY (DEPT_ID)
5  );
6
7  CREATE TABLE OFFICE_INFO (
8   OFFICE_ID INTEGER NOT NULL,
9   OFFICE_NAME VARCHAR(50),
10  HAS_PRINTER SMALLINT,
11  PRIMARY KEY (OFFICE_ID)
12 );
```

Figure: A sample of the UnixUsage schema.

Database Schemas

```
1  CREATE TABLE DEPT_INFO (
2   DEPT_ID INTEGER NOT NULL,
3   DEPT_NAME VARCHAR(50),
4   PRIMARY KEY (DEPT_ID)
5  );
6
7  CREATE TABLE OFFICE_INFO (
8   OFFICE_ID INTEGER NOT NULL,
9   OFFICE_NAME VARCHAR(50),
10  HAS_PRINTER SMALLINT,
11  PRIMARY KEY (OFFICE_ID)
12 );
```

Figure: A sample of the UnixUsage schema.

Testing Database Schemas

Manual testing is onerous and error prone

Testing Database Schemas

Manual testing is onerous and error prone

DBMonster only supports one DMBS

Testing Database Schemas

- Manual testing is onerous and error prone
- DBMonster only supports one DMBS
- Crashes and poor constraint coverage

Testing Database Schemas

Manual testing is onerous and error prone

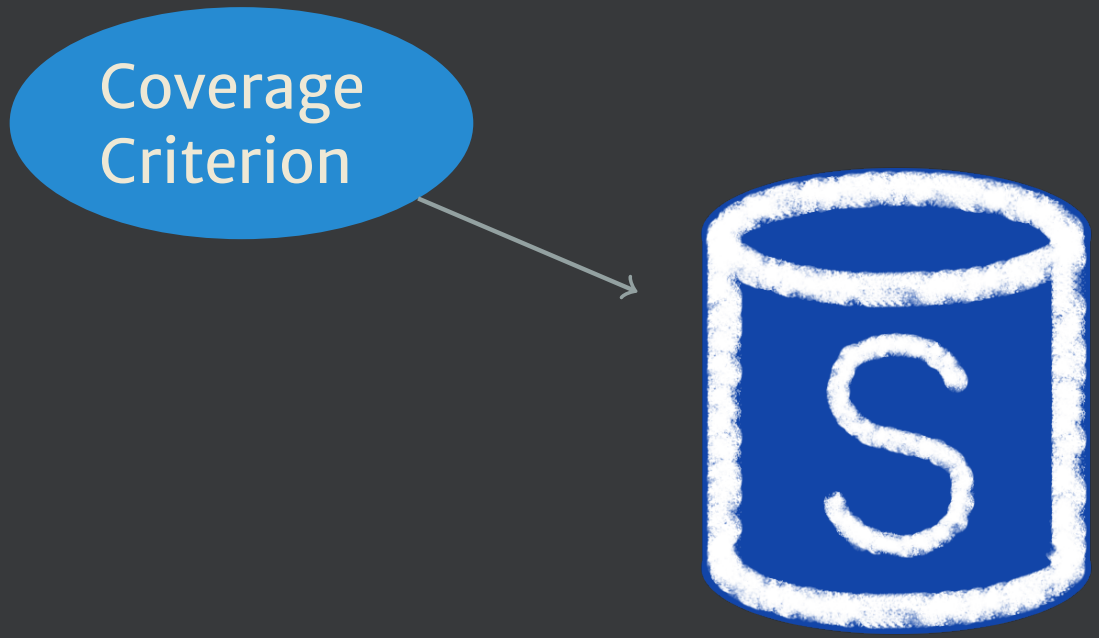
DBMonster only supports one DMBS

Crashes and poor constraint coverage

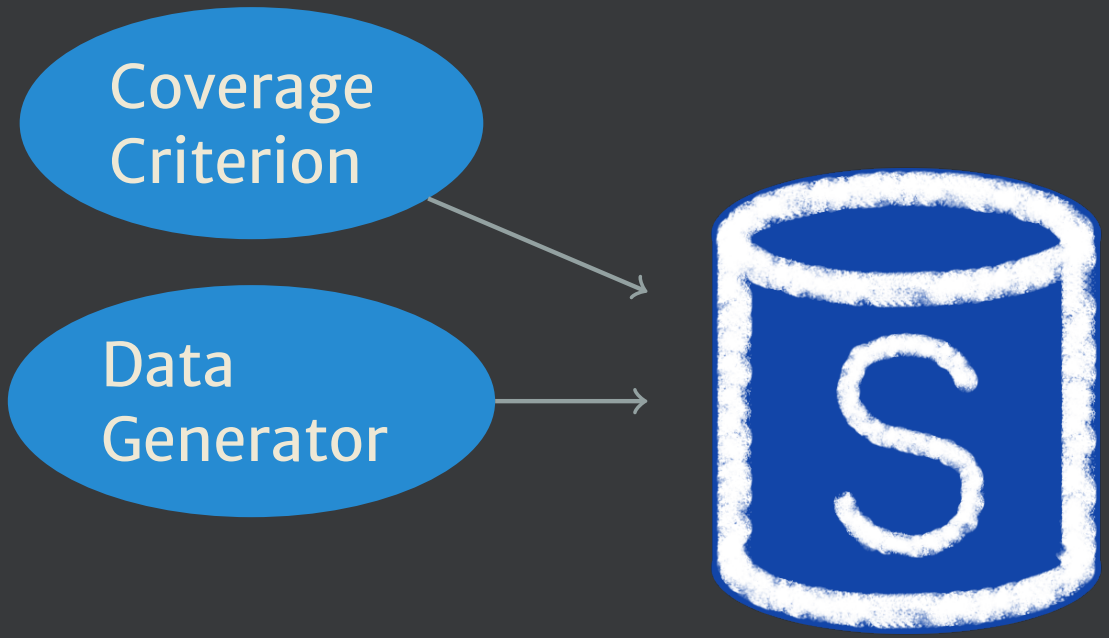
Schemas often not tested at all!

SchemaAnalyst

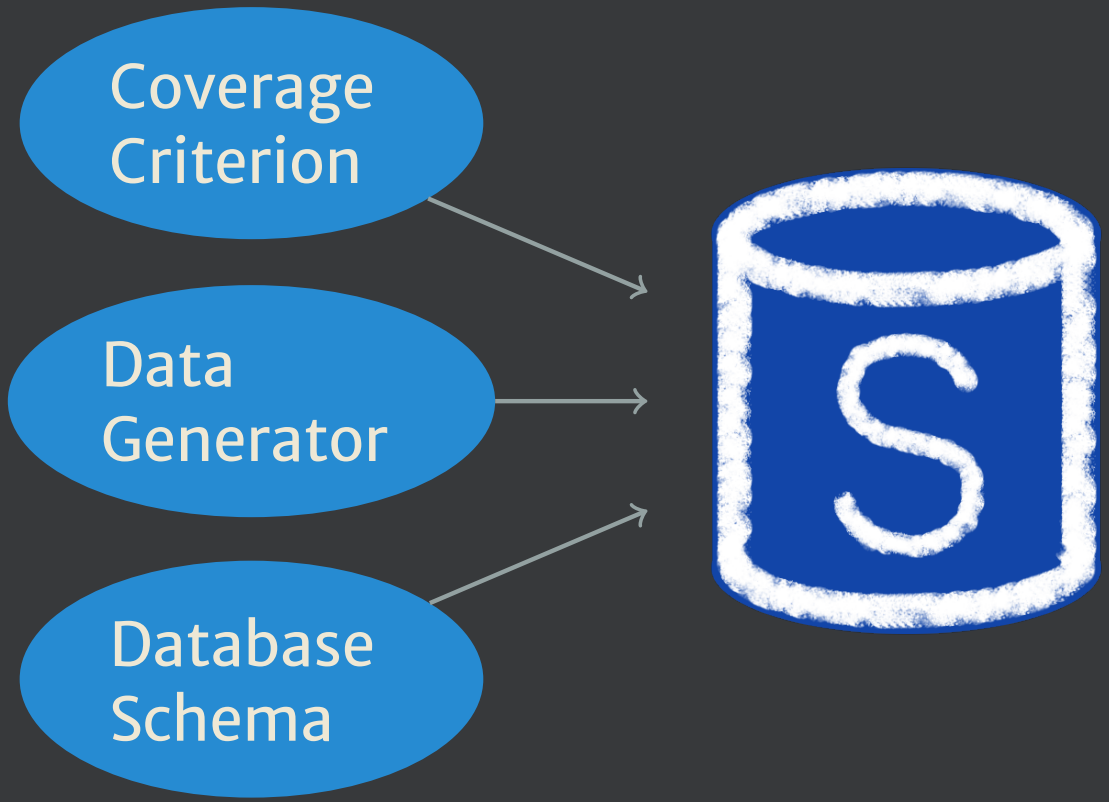
SchemaAnalyst



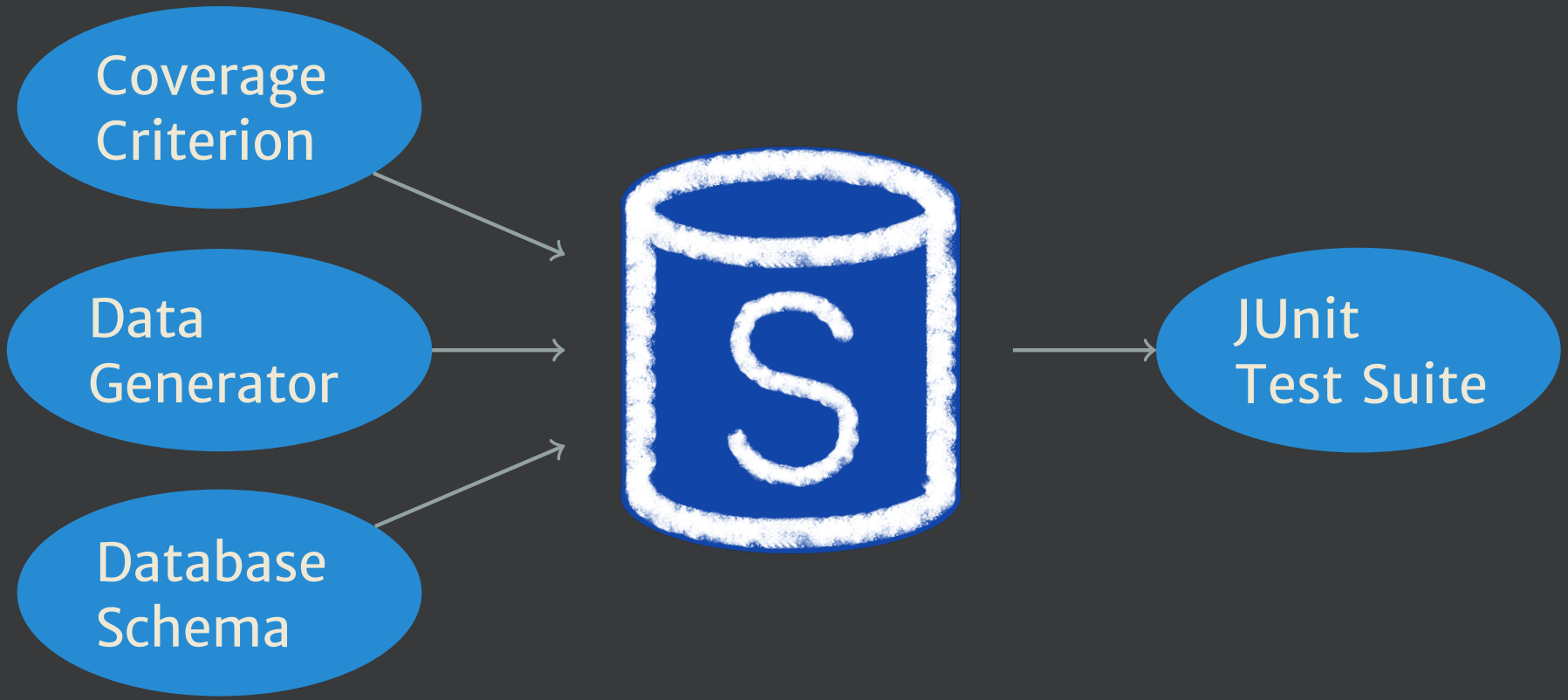
SchemaAnalyst



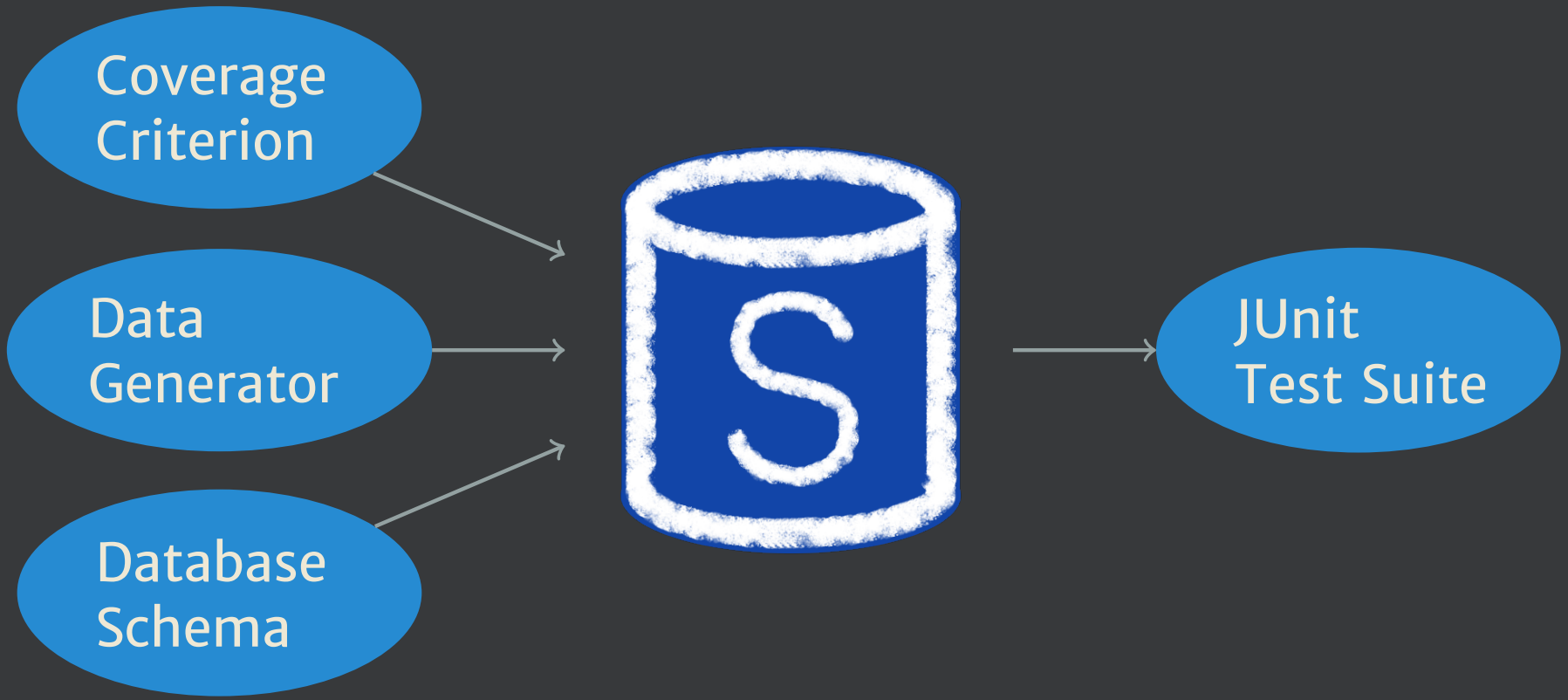
SchemaAnalyst



SchemaAnalyst

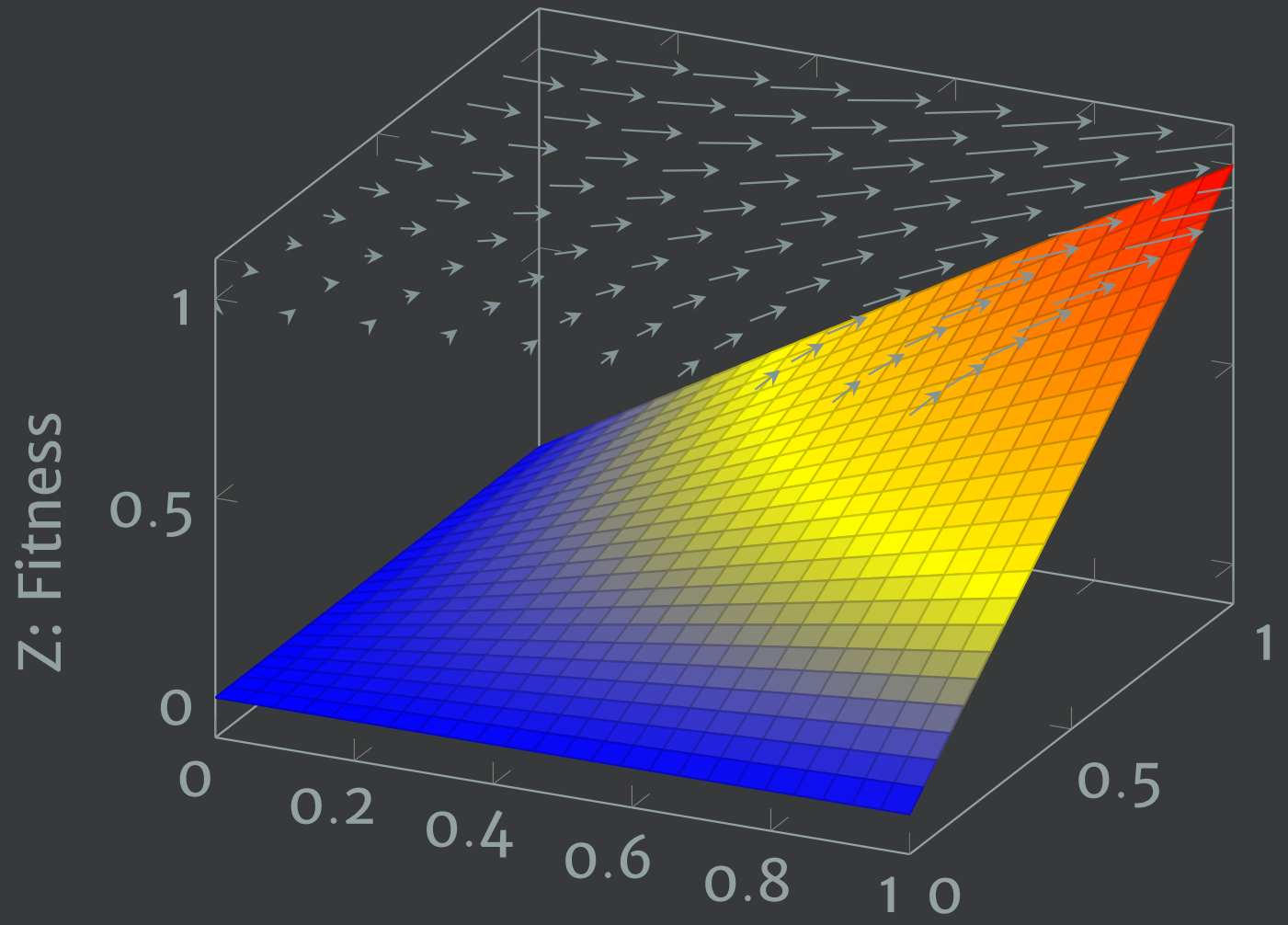


SchemaAnalyst



Extensible tool for test data generation

Search-Based Testing



Automatically Testing Schemas

```
1 CREATE TABLE DEPT_INFO (
2  DEPT_ID INTEGER NOT NULL,
3  DEPT_NAME VARCHAR(50),
4  PRIMARY KEY (DEPT_ID)
5 );
```

Figure: A sample of the UnixUsage schema.

Automatically Testing Schemas

```
1 CREATE TABLE DEPT_INFO (
2  DEPT_ID INTEGER NOT NULL,
3  DEPT_NAME VARCHAR(50),
4  PRIMARY KEY (DEPT_ID)
5 );
```

Figure: A sample of the UnixUsage schema.

```
1 INSERT INTO DEPT_INFO
  VALUES (0, '');

2 INSERT INTO DEPT_INFO
  VALUES (NULL, '');
```

Figure: Data generated by *SchemaAnalyst*.

Automatically Testing Schemas

```
1 CREATE TABLE DEPT_INFO (
2  DEPT_ID INTEGER NOT NULL,
3  DEPT_NAME VARCHAR(50),
4  PRIMARY KEY (DEPT_ID)
5 );
```

Figure: A sample of the UnixUsage schema.

```
1 INSERT INTO DEPT_INFO
  VALUES (0, '');

2 INSERT INTO DEPT_INFO
  VALUES (NULL, '');
```

Figure: Data generated by *SchemaAnalyst*.

Automatically Testing Schemas

```
1  CREATE TABLE DEPT_INFO (
2   DEPT_ID INTEGER NOT NULL,
3   DEPT_NAME VARCHAR(50),
4   PRIMARY KEY (DEPT_ID)
5  );
```

Figure: A sample of the UnixUsage schema.

```
1  INSERT INTO DEPT_INFO
   VALUES (0, '');

2  INSERT INTO DEPT_INFO
   VALUES (NULL, '');
```

Figure: Data generated by *SchemaAnalyst*.

Automatically Testing Schemas

```
1 CREATE TABLE DEPT_INFO (
2  DEPT_ID INTEGER NOT NULL,
3  DEPT_NAME VARCHAR(50),
4  PRIMARY KEY (DEPT_ID)
5 );
```

Figure: A sample of the UnixUsage schema.

```
1 INSERT INTO DEPT_INFO
  VALUES (0, '');

2 INSERT INTO DEPT_INFO
  VALUES (NULL, '');
```

Figure: Data generated by *SchemaAnalyst*.

Automatically Testing Schemas

```
1  CREATE TABLE DEPT_INFO (
2   DEPT_ID INTEGER NOT NULL,
3   DEPT_NAME VARCHAR(50),
4   PRIMARY KEY (DEPT_ID)
5  );
```

Figure: A sample of the UnixUsage schema.

```
1  INSERT INTO DEPT_INFO
   VALUES (0, '');

2  INSERT INTO DEPT_INFO
   VALUES (NULL, '');
```

Figure: Data generated by *SchemaAnalyst*.

Automatically Testing Schemas

```
1  CREATE TABLE DEPT_INFO (
2   DEPT_ID INTEGER NOT NULL,
3   DEPT_NAME VARCHAR(50),
4   PRIMARY KEY (DEPT_ID)
5  );
```

Figure: A sample of the UnixUsage schema.

```
1  INSERT INTO DEPT_INFO
   VALUES (0, '');

2  INSERT INTO DEPT_INFO
   VALUES (NULL, '');
```

Figure: Data generated by *SchemaAnalyst*.

Automatically Testing Schemas

```
1  CREATE TABLE DEPT_INFO (
2   DEPT_ID INTEGER NOT NULL,
3   DEPT_NAME VARCHAR(50),
4   PRIMARY KEY (DEPT_ID)
5  );
```

Figure: A sample of the UnixUsage schema.

```
1  INSERT INTO DEPT_INFO
   VALUES (0, '');

2  INSERT INTO DEPT_INFO
   VALUES (NULL, '');
```

Figure: Data generated by *SchemaAnalyst*.

Automatically Testing Schemas

```
1  CREATE TABLE DEPT_INFO (
2   DEPT_ID INTEGER NOT NULL,
3   DEPT_NAME VARCHAR(50),
4   PRIMARY KEY (DEPT_ID)
5  );
```

Figure: A sample of the UnixUsage schema.

```
1  INSERT INTO DEPT_INFO
   VALUES (0, '');

2  INSERT INTO DEPT_INFO
   VALUES (NULL, '');
```

Figure: Data generated by *SchemaAnalyst*.

Automatically Testing Schemas

```
1  CREATE TABLE DEPT_INFO (
2   DEPT_ID INTEGER NOT NULL,
3   DEPT_NAME VARCHAR(50),
4   PRIMARY KEY (DEPT_ID)
5  );
```

Figure: A sample of the UnixUsage schema.

```
1  INSERT INTO DEPT_INFO
   VALUES (0, '');

2  INSERT INTO DEPT_INFO
   VALUES (NULL, '');
```

Figure: Data generated by *SchemaAnalyst*.

Automatically Testing Schemas

```
1  CREATE TABLE DEPT_INFO (
2   DEPT_ID INTEGER NOT NULL,
3   DEPT_NAME VARCHAR(50),
4   PRIMARY KEY (DEPT_ID)
5  );
```

Figure: A sample of the UnixUsage schema.

```
1  INSERT INTO DEPT_INFO
   VALUES (0, '');

2  INSERT INTO DEPT_INFO
   VALUES (NULL, '');
```

Figure: Data generated by *SchemaAnalyst*.

Automatically Testing Schemas

```
1  CREATE TABLE DEPT_INFO (
2   DEPT_ID INTEGER NOT NULL,
3   DEPT_NAME VARCHAR(50),
4   PRIMARY KEY (DEPT_ID)
5  );
```

Figure: A sample of the UnixUsage schema.

```
1  INSERT INTO DEPT_INFO
   VALUES (0, '');

2  INSERT INTO DEPT_INFO
   VALUES (NULL, '');
```

Figure: Data generated by *SchemaAnalyst*.

Automatically Testing Schemas

```
1  CREATE TABLE DEPT_INFO (
2   DEPT_ID INTEGER NOT NULL,
3   DEPT_NAME VARCHAR(50),
4   PRIMARY KEY (DEPT_ID)
5  );
```

Figure: A sample of the UnixUsage schema.

```
1  INSERT INTO DEPT_INFO
   VALUES (0, '');

2  INSERT INTO DEPT_INFO
   VALUES (NULL, '');
```

Figure: Data generated by *SchemaAnalyst*.

Automatically Testing Schemas

```
1  CREATE TABLE DEPT_INFO (
2   DEPT_ID INTEGER NOT NULL,
3   DEPT_NAME VARCHAR(50),
4   PRIMARY KEY (DEPT_ID)
5  );
```

Figure: A sample of the UnixUsage schema.

```
1  INSERT INTO DEPT_INFO
   VALUES (0, '');

2  INSERT INTO DEPT_INFO
   VALUES (NULL, '');
```

Figure: Data generated by *SchemaAnalyst*.

Automatically Testing Schemas

```
1  CREATE TABLE DEPT_INFO (
2   DEPT_ID INTEGER NOT NULL,
3   DEPT_NAME VARCHAR(50),
4   PRIMARY KEY (DEPT_ID)
5  );
```

Figure: A sample of the UnixUsage schema.

```
1  INSERT INTO DEPT_INFO
   VALUES (0, '');

2  INSERT INTO DEPT_INFO
   VALUES (NULL, '');
```

Figure: Data generated by *SchemaAnalyst*.

Real-world Ready

Schemas from Firefox and StackOverflow

Real-world Ready

- Schemas from Firefox and StackOverflow
- Scales to 1,000s of tables and constraints

Real-world Ready

Schemas from Firefox and StackOverflow

Scales to 1,000s of tables and constraints

Extensive documentation available on GitHub

Real-world Ready

Schemas from Firefox and StackOverflow

Scales to 1,000s of tables and constraints

Extensive documentation available on GitHub

SchemaAnalyst provides an efficient means of generating test data for real-world database applications

Usage

Tool Demo

Key Contributions

SchemaAnalyst: an open-source test data generator for relational database schemas

Key Contributions

SchemaAnalyst: an open-source test data generator for relational database schemas

Extensible to new data generators, coverage criteria, and database management systems

Key Contributions

SchemaAnalyst: an open-source test data generator for relational database schemas

Extensible to new data generators, coverage criteria, and database management systems

Extensive documentation support-ing the use and modification of the tool

Key Contributions

SchemaAnalyst: an open-source test data generator for relational database schemas

Extensible to new data generators, coverage criteria, and database management systems

Extensive documentation supporting the use and modification of the tool

Enhance the testing of database systems in industry and enable future research!

Key Contributions

SchemaAnalyst: an open-source test data generator for relational database schemas

Extensible to new data generators, coverage criteria, and database management systems

Extensive documentation supporting the use and modification of the tool

Enhance the testing of database systems in industry and enable future research!

https://github.com/schemaanalyst-team/schemaanalyst